Megerősítéses tanulás és MCTS alkalmazása trajektóriakövetésben

Kővári Bálint * Bécsi Tamás*
Szalay Zsolt**

** Közlekedés- és Járműirányítási Tanszék, Budapesti Műszaki és Gazdaságtudományi Egyetem*
(e-mail: balint.kovary@gmail.com; becsi.tamas@mail.bme.hu)
*** Gépjárműtechnológia Tanszék, Budapesti Műszaki és Gazdaságtudományi Egyetem*
(e-mail: zsolt.szalay@gjt.bme.hu)

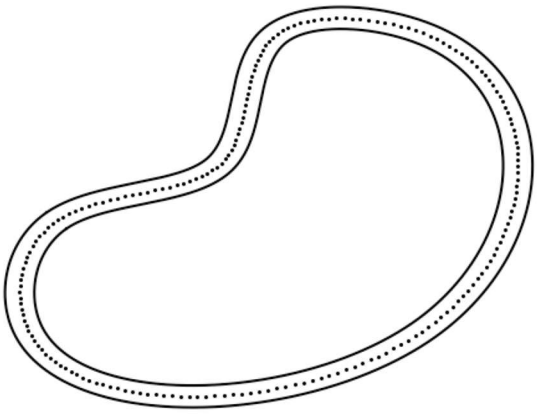
Kivonat: A publikáció bemutatja a mesterséges intelligencia új területét megtestesítő megerősítéses tanulás -Q-learning és Policy Gradient- és klasszikus keresés és tervezés alapú megközelítéseket, járműirányítási feladatok megoldására. A felvetés célja, hogy megtaláljuk az eredmények összehasonlításán keresztül az egyes irányzatok alkalmazásspecifikus előnyeit és hátrányait. A feladat egy kinematikai bicikli modell trajektória követésének megvalósítása. A probléma absztrahálásához magasszintű szenzorinformációkat használunk, a trajektória követési feladat meghatározásához pedig egy kiértékelő függvényt hozunk létre, amely a jármű helyzetét vizsgálja a pálya viszonylatában minden egyes mintavételezés során. Az algoritmusok eredményeinek összehasonlítását, a trajektória követés stabilitása és annak minősége alapján valósítjuk meg.

1. BEVEZETÉS

Az autóipar szempontjából, kiemelt figyelmet élvez a minél magasabb szintű automatizációt megvalósító önvezető rendszerek elérése [1]. Ennek megfelelően az újabb modellekben már találkozhatunk vezetéstámogató, illetve önvezető funkciókkal is, azonban ezek a már hozzáférhető rendszerek csupán 2-es szintet érik el. A fejlesztések legfőbb gátja a közlekedési környezet és szituációk sokfélesége és komplexitása, ez egyben a legfőbb oka, hogy klasszikus algoritmusok használata mellett egyre nagyobb hangsúlyt kapnak a mesterséges intelligencia tárgykörébe tartozó megoldások is. Járműirányítási feladatokra gyakran felügyelt tanulás alapú megoldásokat alkalmaznak, melyek mesterséges neurális hálózatokat használnak [2]. Ebben az esetben a szabályozáshoz használt bemenet egy kamera által rögzített képsorozat, ezeknek az algoritmusoknak a legfőbb hátránya, hogy tanulás során nagy mennyiségű címkézett adatra van szükségük, melyek gyakran nem állnak rendelkezésre vagy rendkívül költséges az előállításuk. Ezért is fontos a gépi tanulás egy másik területe, a megerősítéses tanulás, ami napjainkban rengeteg fejlesztő és kutató figyelmét nyerte el, melynek legfőbb oka a Google Deepmind által elért eredmények, mint például [3] vagy [4]. Ezek az eredmények többnyire videójátékok, egyszerűbb szabályozási feladatok és táblajátékok területéről származnak. Fontos előnye az előbb említett felügyelt tanulással szemben, hogy nem igényel nagy mennyiségű tanító adathalmazt. Az említett eredmények közül különös érdeklődés övezi a [5]-t, önmagában a Go-ban elért eredmény is szignifikáns, de talán legérdekesebb, hogy az ágens olyan stratégiát talált és használt, amely nem emberi, tehát új összefügéseket ismert fel amelyek az emberitől különböző gondolkodásmódot közvetítenek. Ez is egy fontos különbséget mutat a két fajta tanulás között. Felügyelt esetben korlátozott az elérhető eredmény, hiszen a feladat célja, hogy rekonstruáljuk a bemenet és a kimenet közötti kapcsolatot, ez a korlátozás nem vonatkozik a megerősítéses tanulásra. A gépi tanulás algoritmusainak gyengesége, hogy megbízhatóságukra nincs garancia, csupán valószínűség, ennek legfőbb oka, hogy kizárólag abban a környezetben tudnak megfelelően működni, melyet tanulás során reprezentáltunk számukra. Ezt a veszélyt legjobban az un. ellenséges példák mutatják be [6].
A publikáció célja, hogy a gépi tanulás alapú algoritmusok és klasszikus algoritmusok, trajektória követési feladat során nyújtott teljesítményének összehasonlításán keresztül megvizsgáljuk, az [5]-hez hasonló megoldások megvalósíthatóságában rejlő potenciált járműirányítási feladatok területén.

1.1 A környezet

Legfontosabb elvárások a környezettel szemben a rugalmasság és testreszabhatóság, ezekre azért van szükségünk, hogy az egyes algoritmusok által szolgáltatott megoldások minőségét megfelelően felmérhessük és azokat összehasonlíthassuk. A környezet lehetővé teszi, hogy tetszőleges pályát generálhassunk annak tartópontjai segítségével, illetve a pályát alkotó pontok száma is előre megválasztható. Fontos továbbá a környezet vizualizációja mely nagyban segíti az esetleges hibák keresését, ehhez egy változtatható szélességű sávot hoztunk létre melyet a valódi pályagörbe oszt ketté.
A környezet diszkrét idejű, a mintavételezési idő pedig tetszőlegesen megválasztható. Az 1.ábra egy a tanítás során használt pályát szemlélteti, a pálya pontok és a hibahatárt jelző sáv segítségével.



1.Ábra. A környezet vizualizációja

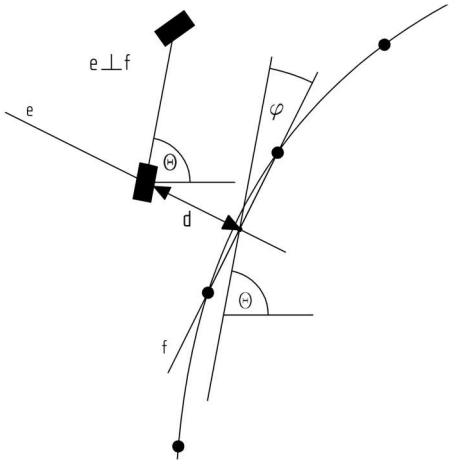
1.2 Szenzorinformációk

A probléma megoldásának kulcsfontosságú része azoknak az információknak az megválasztása és előállítása, amelyek a feladat reprezentálásáért felelősek. A célunk az, hogy olyan tömör megfogalmazást alakítsunk ki, amely önmagában felgyorsítja és robusztussá teszi a megoldást. Trajektória követés esetén ezt úgy érhetjük el, ha kizárólag relatív -tehát csak a pálya és a jármű közötti viszony által befolyásolt- információkat használunk. Ebből kifolyólag két fő részre bonthatjuk a szenzorinformációkat tartalmazó vektort, hiszen a reprezentáció célja is kettős, egyrészt fontos a jármű helyzetének meghatározása a pálya függvényében, másrészt szükségünk van olyan információkra melyek a jármű aktuális állapota alapján írják le a pálya változását egy előre meghatározott mértékű előre tekintés esetén. A bemutatott két komponens azért teljes mértékben egyenrangú, mert a megfelelő minőségű trajektória követés megkívánja, hogy az aktuális beavatkozás mind az adott mintavételezésnek mind a pálya későbbi alakulásának megfeleljen. A 2.ábra bemutatja a szenzorinformációk számítási elvét és a felhasznált paramétereket.

1.Táblázat Szenzorvektor összetétele

d	φ_0	$\varphi_{2.5m}$	$\varphi_{7.5m}$	φ_{15m}	φ_{30m}
[%]	[rad]	[rad]	[rad]	[rad]	[rad]
Távolság	Állásszög	Átlagolt állásszögek			

Az 1.táblázat a feladat absztrahálásához használt szenzorvektor összetételét szemlélteti, amely a tanítás során használt neurális hálózat bemenete is egyben. A táblázatban bemutatott szögek mindegyikét $\pm\pi$ közé normáljuk, majd egyenként normalizáljuk őket. Ennek az a legfőbb oka, hogy ezáltal a teljes szenzorinformáció vektor a $[-1,+1]$ értéktartományba kerül, aminek különösen fontos jelentősége van, a tanítás stabilitásának megőrzésében és a neurális hálózatok egyik legnagyobb gyengepontja a numerikus problémák elkerülésében. Továbbá az értéktartomány szempontjából nem egyenletes reprezentáció az optimalizációs feladatot is negatívan befolyásolja.

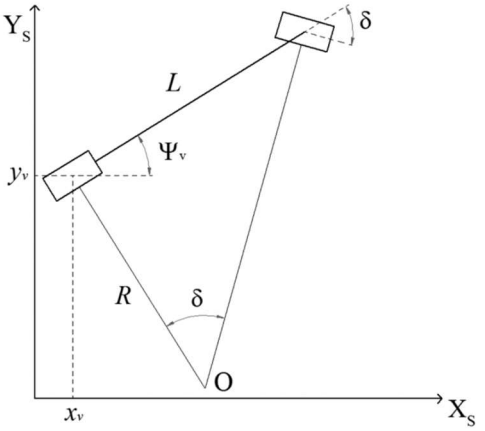


2.Ábra. Szenzorinformációk értelmezése

Az így létrehozott szenzorinformációk lehetővé teszik, hogy tanítás során minden olyan trajektóriát megtanuljon a döntéshozó, amely a pályán használt kanyarok görbületéből adódik ez jelentős robusztusságot kölcsönöz azzal a megközelítéssel szemben, ahol csak az adott trajektóriát tanulják meg az abszolút információk hatására. Ez tekinthető a tanítási folyamat implicit felgyorsításának is.

1.3 Járműmodell

Az alkalmazott járműmodell egy merev kinematikai bicikli modell, ahol a laterális irányba történő elmozdulást kizárólag a geometriai paraméterek befolyásolják. Tehát elhanyagoljuk a kerék slip hatását. Az említett modellt a 3.ábra szemlélteti.



3.Ábra. Kinematikai bicikli modell

Ezáltal a pálya R sugara (1) szerint adódik.

$$\tan\delta = \frac{L}{R} \tag{1}$$

A szögsebesség pedig (2) alapján számítható.

$$\dot{\psi} = \frac{v}{R} = \tan\delta\frac{v}{L} \tag{2}$$

A felhasznált kinematikai bicikli modell pontos leírása [7]-ben található.

1.4 Beavatkozás típusok

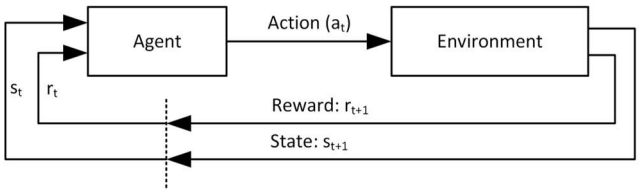
A jármű irányításához, kizárólag előre meghatározott kormányszögeket használunk, melyeket szimmetrikusan választottunk meg. A szögek meghatározása során fontos szempont volt, hogy ne legyen túlsok kormányszög, hiszen az a tanulás alapú és a keresés alapú megoldásokat is hátrányosan érintené a nagyobb számítási igénye miatt. Továbbá fontos kiemelni, hogy a jármű sebessége rögzített.

2. ALGORITMUSOK

A megoldáshoz használt algoritmusok alapvetően két csoportba oszthatók. Az elsőben megerősítéses tanulás alapú megközelítést alkalmazunk, míg a másikban előre tekintő keresés és tervezés segítségével alakítjuk ki az aktuális pályához tartozó irányítást a jármű számára.

2.1 Megerősítéses Tanulás alapú megoldások

A megerősítéses tanulás esetében a döntéshozót ágensnek nevezzük. A tanulási folyamat gyakorlatilag az ágens és a környezet közötti interakciók egymásutánját jelenti. A környezet feladata az ágens számára előállítani azokat az információkat, amelyek alapján a döntéshozás megvalósítható, illetve egy skalár visszacsatolásával értékelni az ágens által választott beavatkozás következményeit, mely lehet jutalom amennyiben pozitív és lehet büntetés amennyiben negatív. Tanulás során az ágens által használt függvény approximátor - itt egy neurális hálózat- paramétereit hangoljuk az ágens döntései és a környezet visszacsatolásai által meghatározott veszteség függvény segítségével. A tanulási folyamatot epizódokra bontjuk, mert több olyan állapottal is rendelkezik a környezet, melyek elérése esetén a folyamatot megszakítjuk és a kezdeti állapotba való visszaállás után újraindítjuk. Minden olyan állapot végső állapot, ahol a jármű elhagyja a trajektória követés céljából létrehozott sávot vagy ha a pálya végére jut. A 4.ábra a megerősítéses tanulás elvét szemlélteti.



4.Ábra: Interakció az ágens és a környezet között.

Jelölések:

- s_t: Szenzorinformáció vektor a t-edik mintavételezésben
- r_t: A környezet visszacsatolása a t-edik mintavételezésben
- a_t: Az ágens által választott beavatkozás a t-edik mintavételezésben

Az egyetlen mennyiség, amely a választott beavatkozás minőségét mutatja a környezet által szolgáltatott visszacsatolás, ezért ezt úgy kell létrehozni, hogy nagyon pontos képet adjon a trajektória követés minőségéről, ennek megfelelően, ha a visszacsatolás interakciók során történő alakulását vizsgáljuk a trajektória követés stabilitásáról kaphatunk képet. A visszacsatolás generálásának elvét az (3) szemlélteti.

$$r_t = \cos\varphi_0 - |d| \tag{3}$$

Az (3) egyenlet jelentése egyszerűen annyi, hogy egy tetszőleges állapot akkor a legjobb, ha a jármű párhuzamos az adott pályaponthoz tartozó érintővel és a sávközéptől vett távolsága zérus. Innen adódik az ágens feladata, ami a kumulatív visszacsatolás maximalizálása a teljes epizódra nézve. A kumulatív visszacsatolás (4) alapján adódik.

$$G = \sum_{t=0}^{\tau} \gamma^t r_t \tag{4}$$

Változók:

- γ^t: Discount factor

A discount factor meghatározza mennyiben függenek a jövőbeli visszacsatolások a múltbéliektől ($0 \leq \gamma \leq 1$). A (4) segítségével meghatározható egy tetszőleges állapot értéke ezt (5) szemlélteti.

$$V_\pi = E_\pi[G|s_t] \tag{5}$$

A (5) alapján pedig belátható, hogy létezik egy optimális viselkedés π^* ahol $V^{\pi*}$ optimális minden állapothoz. Tehát egy tömörebb megfogalmazásban az ágens célja a π^* viselkedés megtalálása.

2.1.1 Q-learning

Ennek a megközelítésnek a lényege, hogy nem az állapotokhoz tartozó értékeket tanulja meg az ágens, hanem a Q-függvényt, amely minden állapot lehetséges beavatkozásaihoz hozzárendeli a beavatkozás megvalósításával megszerezhető kumulatív visszacsatolást. A Q-függvényt a Bellman egyenletek segítségével frissítjük.

$$Q(s_t, a_t) = r_t + \gamma \max(Q(s_{t+1}, a_t)) \tag{6}$$

Tehát a cél a Q-függvény minél pontosabb becslése a neurális hálózat segítségével. Fontos megjegyezni, hogy ez gyakorlatilag egy indirekt módszer az optimális viselkedés megtanulására ugyanis a prediktált Q-értékek nem javasolnak semmilyen viselkedést, inkább egyfajta helyzetértékelést szolgáltatnak az aktuális állapothoz, az ágensen múlik hogyan használja fel ezt az információt. Továbbá a Q-függvény ilyen módon való frissítéséből adódik, hogy ez egy off-policy módszer, hiszen frissítéshez nem azt a Q-értéket használjuk amelyik a megvalósított beavatkozáshoz tartozik. Tehát frissítéskor az egyenlet nem azt veszi figyelembe, ami valóban

történt, hanem ami a legjobb lett volna, ha a történik. A megerősítéses tanulás ezen ágának fő problémája, hogy nem garantált a konvergencia, azonban tovább fejlesztett algoritmusokkal gyakorlatban jó eredményeket lehet elérni [8].

2.1.2 Policy Gradient

A megerősítéses tanulás másik fő típusa a Policy Gradient módszer, amit napjainkban különös érdeklődés övez szabályozási problémák területén mutatott bíztató eredményei miatt [9]. A legfontosabb különbség a bemutatott Q-learning-hez képest, hogy itt direkt módon tanulja az ágens az optimális viselkedést, illetve ez az algoritmus garantáltan konvergál viszont néhány esetben a globális optimum helyett lokális optimumot felé [10]. Ebben az esetben az ágens kimenete egy valószínűségi eloszlás a lehetséges beavatkozásokra nézve $\pi_\theta(a_t|s_t)$ amit a függvény approximátor θ paraméterei határoznak meg. Ebből kifolyólag egy optimalizációs feladatot kapunk melynek célja a θ paraméterek meghatározása úgy, hogy közben az maximalizálja a $J(\theta) = J(\pi_\theta)$ függvényt, amely az ágens teljesítményének indikátora és epizódikus környezetekben (7) szerint határozható meg.

$$J(\pi_\theta) = \mathbf{E}\left[\sum_{t=0}^{T} r_t\right] \tag{7}$$

Az algoritmus lényege, hogy $J(\theta)$ lokális maximumát keresi a kumulatív visszacsatolás legnagyobb növekedésének irányát követve. Innen adódik az approximátor paramétereinek frissítési szabálya, melyet (8) szemléltet.

$$\Delta\theta = \alpha\nabla_\theta J(\theta) \tag{8}$$

Paraméterek:

- α: Tanulási ráta

A következő fontos lépés a gradiensek számítása, melynek több lehetséges módja ismert. Ebben az esetben a reinforce algoritmus [11] által alkalmazott módszert követtük. A gradiensek így történő meghatározását (9) szemlélteti.

$$\nabla J(\theta) = \mathbf{E}\left[\nabla_\theta log\pi_\theta(s,a)\sum_{t=0}^{T} r_t\right] \tag{9}$$

Így a paraméterek frissítési szabálya (10) módosul.

$$\theta \leftarrow \theta + \alpha\nabla log\pi_\theta(s_t,a_t)\sum_{t=0}^{T} r_t \tag{10}$$

A stabilabb tanulási folyamat érdekében érdemes nem minden epizód végén frissíteni a paramétereket, hanem kiszámolni a gradienseket minden epizódhoz, kumulálni őket majd néhány epizód kihagyás után az így kapott gradiensekkel frissíteni a paramétereket. Azonban ezáltal egy újabb hiperparaméterhez jutunk, ami az approximátor paramétereinek frissítési frekvenciája lesz.

2.2 Megoldás előre tekintő keresés és tervezés segítségével

A keresés alapú megoldások működésének elve, hogy minden mintavételezésben az azt leíró állapotból kiindulva felépítenek egy fát, aminek az elágazási tényezője a lehetséges beavatkozások számával megegyező, mélysége pedig az előre tekintés mértékétől függően változhat. A fa egyes csúcsai lehetséges jövőbeli állapotokat szimbolizálnak az értékük pedig a már ismert (3) alapján számítható. Az egyes csúcsok értékét a fában a szülő felé felpropagálva megkaphatjuk, hogy az aktuális állapotban melyik lépés lehet a legmegfelelőbb adott előre tekintés mellett. A megközelítés legfőbb problémája, hogy a megvizsgálandó csúcsok száma a mélységnövekedésével exponenciálisan növekszik, ami a megoldáshoz szükséges számítási idő, kapacitás és memória nagymértékű növekedésével jár.

2.2.1 Monte Carlo Tree Search

A felvázolt problémát az MCTS algoritmus nagy mértékben mérsékli. Az MCTS algoritmus iránti érdeklődés az elmúlt néhány évben rendkívül megnőtt a Deepmind által több területen is elért sikerek miatt. Az algoritmus célja optimális döntések meghozatala véletlenszerű szimuláció és fakeresés segítségével. Az algoritmus működése során felépít egy keresési fát, csúcsról csúcsra a véletlenszerű szimulációk segítségével. Ez a folyamat négy lépésre bontható:

- ***Csúcs választás***: Az aktuális állapotból indulva rekurzívan kiválasztjuk a legígéretesebbnek tűnő csúcsot addig amig olyan csúcsot nem választunk, amelyből nincsen további elágazás.
- ***Expanzió***: Amennyiben a választott csúcs nem végső állapot meghatározzuk a belőle elágazó csúcsokat majd választunk közülük egyet.
- ***Szimuláció***: A választott csúcsot kezdő állapotként használva végig játszúk a játékot és az így szerzett eredményt a csúcs értékének választjuk.
- ***Visszapropagálás***: A választott csúcshoz vezető útvonalon található csúcsok értékét kiegészítjük a választott csúcs értékével.

Az algoritmus legfontosabb része a csúcs választáshoz használt Upper Confidence Bound (UCB) formula melyet (11) szemléltet [12].

$$v_i + C \times \sqrt{\frac{lnN}{n_i}} \tag{11}$$

Paraméterek:

- v_i: Vizsgált csúcs értéke.
- C: Felfedezés mértékét szabályozó konstans.
- N: A vizsgált csúcsot szülő csúcs vizitjeinek a száma.
- n_i: A vizsgált csúcs vizitjeinek száma.

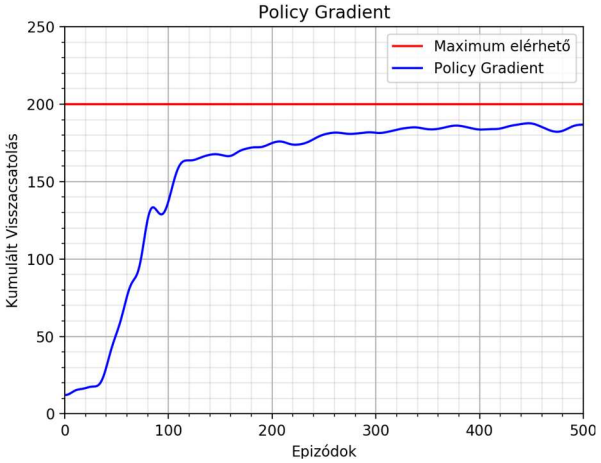
A (11) kifejezés lényegében egy parametrizálható domén specifikus kompromisszum kialakítását teszi lehetővé a greedy és a brute force keresési algoritmusok között, aminek köszönhetően képes egy aszimmetrikus keresési fa felépítésére, illetve segítségével nagy mennyiségű időt és számítási kapacitást és memóriát spórólhatunk meg, azokhoz a keresési módszerekhez képest, amelyek nem használnak domén specifikus információkat a keresési folyamatban.

3. ALGORITMUSOK ÖSSZEHASONLÍTÁSA

Az összehasonlítás során, külön megvizsgáljuk a tanuló algoritmusok által elért eredményeket, illetve összehasonlítjuk az egyes algoritmusokhoz tartozó tanulásifolyamatokat, majd a megerősítéses tanulást alkalmazó módszerek teljesítményét összevetjük a keresés és tervezés segítségével nyert megoldással, végül megvizsgáljuk az egyes módszerek alkalmazhatóságát.

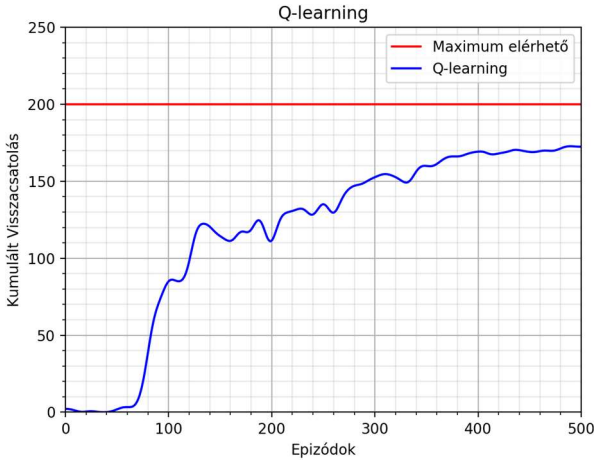
3.1 Tanuló algoritmusok összehasonlítása

A tanulós algoritmusok esetén nagyon fontos minőségjelzők a konvergencia tulajdonságok. Különösképpen meghatározó a konvergencia sebesség és stabilitás. Az 5.ábra a Policy Gradient algoritmus tanulási folyamatát szemlélteti.



5.Ábra. Policy Gradient algoritmus tanítása

A 6.ábrán a Q-learning algoritmus tanulási folyamata látható. Ha a két megoldás főbb konvergencia tulajdonságait összehasonlítjuk, láthatjuk, hogy ebben az esetben a Policy Gradient algoritmus stabilabb és egyben gyorsabban is konvergál. További szempont az algoritmusok lépésre lebontott teljesítményének a vizsgálata. Ebből látható, hogy tanítás után, az egyes megoldások, milyen kilengés mellett képesek az pálya bejárásához tartozó átlag pontszámot megvalósítani.



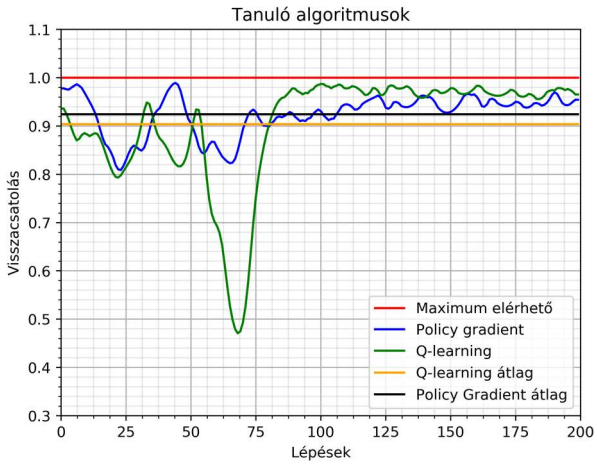
6.Ábra. Q-learning algoritmus tanítása

Ahogy (3) mutatja ez az érték a pálya adott pontjához tartozó érintővel való párhuzamosság és a sávközéptől vett távolság elegyét reprezentálja. A pontosabb analízis érdekében érdemes megnézni az egyes megoldások legnagyobb eltéréseit állásszög és távolság tekintetében, hogy láthassuk, hogyan alakul ki az adott lépéshez tartozó pontszám szélsőséges esetekben.

2.Táblázat Minőségjelzők szélsőértékei

×	Állásszög	Távolság
Policy Gradient	0.18	0.15
Q-learning	0.25	0.47
×	[rad]	[%]

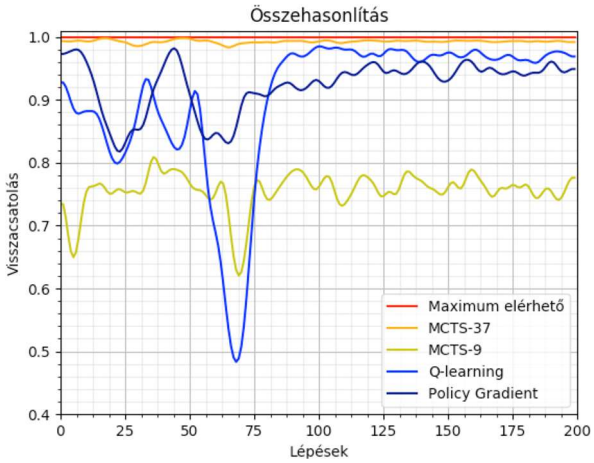
A 2.táblázat alapján az is látható, hogy sokkal dominánsabb a távolság által okozott hiba, mint az állásszög által okozott. Ez azt a jelenti, hogy gyakorlatilag a jármű pályával párhuzamosan halad azonban ezt nem a sávközépen teszi. Továbbá fontos megjegyezni a második táblázat és a 7.ábra tanulságát, ami azt mutatja, hogy ilyen jellegű feladatok esetén a Policy Gradient megbízhatóbb és abszolút értelemben is jobb megoldást ad.



7.Ábra: Az algoritmusok teljesítménye lépésenként

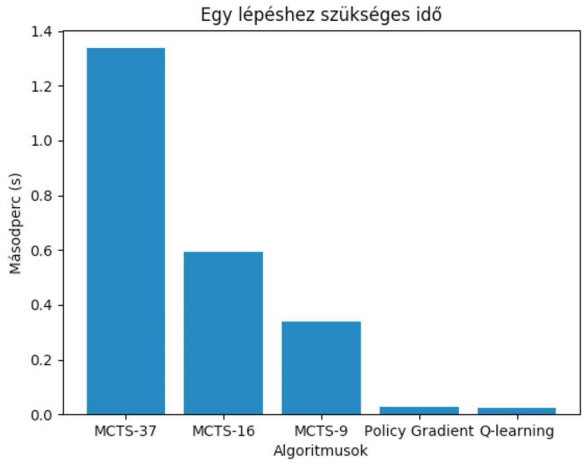
3.2 A két megoldás típus összehasonlítása

A két megoldás típus összevetését a 8.ábra mutatja.



8.Ábra: A két megoldás típus összehasonlítása

Mielőtt jobban megvizsgálnánk az eredményeket meg kell jegyeznünk, hogy az egyes MCTS eredmények különböző mértékű előre tekintésre vonatkoznak, illetve a nagymértékű előre tekintéshez azért nem a tökéletes megoldás tartozik, mert a diszkrét kormányszögek alapvetően hordoznak magukban egy hibát a trajektória követésre nézve. A 8.ábra tanulsága, hogy az tanuló algoritmusokkal elérhető eredmény a nagy és kis mértékű előre tekintéssel elérhető eredmény közé esik és csak kis mértékben marad el a nagy előre tekintéssel rendelkezőtől.



9.Ábra: MCTS algoritmus különböző előre tekintések esetén

Fontos különbség a megoldások teljesítményén túl, azok alkalmazhatósága. A keresés és tervezés alapú megoldások nagy hátránya, hogy a jobb minőségű eredmény érdekében növelnünk kell az előretekintés mértékét, ami az algoritmusok fejlődése ellenére drasztikusan megnöveli az egy döntés meghozatalához szükséges időt, ami komoly probléma, hiszen egy valódi járművel nagy sebességgel haladva, a biztonságos beavatkozás megválasztására és végrehajtására csupán néhány tized másodperc áll rendelkezésre. Ezt az elvárást a tervezést és keresést alkalmazó megoldások nem tudják teljesíteni. Ez a legjelentősebb előnye a tanuló algoritmusoknak. Az egyes megoldások időszükségletét a 9.ábra szemlélteti.

6. KONKLÚZIÓ ÉS FEJLESZTÉSI JAVASLATOK

A legfontosabb különbség a két megoldás típus között, hogy a megerősítéses tanulás alapú megközelítések robusztussága és minősége elmarad az előre tekintő keresés és tervezést alkalmazótól, azonban ebben az esetben úgy érünk el jobb megoldást, hogy működés során rendelkezésünkre áll a modell, illetve semmilyen kikötést nem fogalmazunk meg a futási idő és a felhasználható számítási kapacitással szemben. Mivel egy ilyen algoritmus többnyire beágyazott környezetben működne, ahol az erőforrások korlátozottak, ezért a valóságban kevéssé használható. Tehát valódi perspektíva a tanuló ágensek robusztusságának növelésében rejlik, ehhez [5]-hez hasonlóan a két megközelítés egy architektúrába való integrálása szükséges. Ebből kifolyólag az ágens által használt neurális hálózat fejlesztésre szorul, mert ha a két algoritmushoz, külön-külön hálózatot alkalmazunk szinte garantált az eltérő konvergencia sebesség, ahogy azt a 5.ábra és 6.ábra mutatja.

További fontos fejlesztési irányvonal a lehetséges beavatkozások kibővítése és a járműmodell fejlesztése. Beavatkozások tekintetében fontos lenne a diszkrét kormányszögeket folytonos kormányszögekre cserélni és lehetőséget adni a jármű sebességének a változtatására folytonos gyorsulás/lassulás értéteke segítségével, ez első sorban az ágens gyökeres módosítását igényli. További fejlesztés lehet a kinematikus bicikli modell dinamikusra cserélése ezáltal lépést téve a valóságos körülmények reprodukálása felé.

KÖSZÖNETNYILVÁNÍTÁS

EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén - A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

HIVATKOZÁSOK

[1] SAE International, Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles, 2016.
[2] Z. Chen and X. Huang, “End-to-end learning for lane keeping of selfdriving cars,” in 2017 IEEE Intelligent Vehicles Symposium (IV), June 2017, pp. 1856–1860.
[3] V. e. a. Mnih, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529 EP –, Feb 2015.
[4] D. e. a. Silver, “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, pp. 484 EP – , Jan 2016, article.
[5] ——, “Mastering the game of go without human knowledge,” Nature, vol. 550, pp. 354 EP –, Oct 2017.

[6] Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems.
[7] O. Töro, T. Bécsi, and S. Aradi, "Design of lane keeping algorithm ´ of autonomous vehicle," Periodica Polytechnica. Transportation Engineering, vol. 44, no. 1, p. 60, 2016.
[8] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," CoRR, vol. abs/1511.06581, 2015.
[9] S. Aradi, T. Bécsi, and P. Gáspár, "Policy gradient based reinforcement ´ learning approach for autonomous highway driving," in 2nd IEEE Conference on Control Technology and Applications, 2018.
[10] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," Advances in Neural Information Processing Systems, vol. 12, pp. 1057–1063, 2000.
[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine Learning, vol. 8, pp. 229–256, 1992.
[12] Kocsis, L., & Szepesvári, C. (2006, September). Bandit based monte-carlo planning. In European conference on machine learning (pp. 282-293). Springer, Berlin, Heidelbe